

[illegible][illegible]

```
SSSSSSSS  UU      UU  MM      MM  EEEEEEEEE  DDDDDDDD  IIIIII  TTTTTTTTT
SSSSSSSS  UU      UU  MM      MM  EEEEEEEEE  DDDDDDDD  IIIIII  TTTTTTTTT
SS      UU      UU  MMMM  MMMM  EE      DD      DD  II      TT
SS      UU      UU  MMMM  MMMM  EE      DD      DD  II      TT
SS      UU      UU  MM      MM  EE      DD      DD  II      TT
SSSSSS  UU      UU  MM      MM  EEEEEEE  DD      DD  II      TT
SSSSSS  UU      UU  MM      MM  EEEEEEE  DD      DD  II      TT
SS      UU      UU  MM      MM  EE      DD      DD  II      TT
SS      UU      UU  MM      MM  EE      DD      DD  II      TT
SS      UU      UU  MM      MM  EE      DD      DD  II      TT
SSSSSSSS  UUUUUUUUU  MM      MM  EEEEEEEEE  DDDDDDDD  IIIIII  TT
SSSSSSSS  UUUUUUUUU  MM      MM  EEEEEEEEE  DDDDDDDD  IIIIII  TT
                                                                ....
                                                                ....
                                                                ....
                                                                ....
```

```
LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS
```

(2)	56	DECLARATIONS
(3)	121	TPARSE
(4)	260	SUMSINIT
(5)	351	GET_IS_BLK
(6)	385	PROCESS_FILE
(7)	433	SET_UP_NODES
(8)	498	INSERT_NODE
(9)	540	READ_UPD_LINE
(10)	592	SUMSCINE
(11)	658	LINE_SET
(12)	693	LINE_NUP
(13)	713	LINE_SRC
(14)	739	LINE_UPD
(15)	830	LINE_UPE
(16)	858	LINE_UPR
(17)	887	LINE_BLK
(18)	918	LINE_GET
(19)	980	LINE_EOF
(20)	1000	ACCESS_SRC
(21)	1035	SAVE_SRC_RFA
(22)	1056	RESTORE_SRC_RFA
(23)	1096	ACCESS_UPDATE
(24)	1166	READ_SRC_LINE
(25)	1214	SKIP_SRC_LINES
(26)	1246	COMMAND_CHECK
(28)	1450	SUMSCLOSE


```
0000 1      .TITLE SUM$EDIT
0000 2      .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6
0000 7 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 *  ALL RIGHTS RESERVED.
0000 10
0000 11 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 *  TRANSFERRED.
0000 17
0000 18 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 *  CORPORATION.
0000 21
0000 22 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24
0000 25 *****
0000 26
0000 27
0000 28
0000 29 ++
0000 30 : FACILITY:      SUMSHR shareable library
0000 31
0000 32 : ABSTRACT:
0000 33
0000 34
0000 35 : ENVIRONMENT:  USER MODE
0000 36
0000 37 : AUTHOR:       R. Newland
0000 38
0000 39 : MODIFIED BY:
0000 40
0000 41 :      V03-002 MTR0002      Mike Rhodes      18-May-1983
0000 42 :      Correct handling of file access switching in READ_UPD_LINE
0000 43 :      when an error occurs. Also, make the RAB globally available
0000 44 :      to the TPARSE action routines.
0000 45
0000 46 :      V03-001 MTR0001      Mike Rhodes      19-Jan-1983
0000 47 :      Create and a local UBF for use in SUM$INIT and SUM$LINE.
0000 48 :      The local UBF precludes ACCVIOs resulting from the caller's
0000 49 :      RAB ROP=LOC, when processing SUMSHR's escape character "<".
0000 50
0000 51 :      V02-001      B. Schreiber      21-Mar-1980
0000 52 :      Make totally position independent.
0000 53
0000 54 :--
```

DECLARATIONS

```
0000 56      .SBTTL  DECLARATIONS
0000 57      :
0000 58      :
0000 59      : Macro definitions
0000 60      :
0000 61      DEFUPFBK      : Source update merge offsets
0000 62      DEFEDBLK      : Edit block offsets
0000 63      DEFISBLK      : Input stream block offsets
0000 64      DEFCMDTYPE    : Command line type
0000 65      DEFSUMCBL     : SUM control block
0000 66      $FABDEF       : FAB
0000 67      $RABDEF       : RAB
0000 68      $NAMDEF       : NAM block
0000 69      $TPADEF       : TPARSE definitions
0000 70      $RMSDEF       : RMS definitions
0000 71      :
0000 72      :
0000 73      : state definitions
0000 74      :
0000 75      $EQU LST SUM_ST,,0,,< -
0000 76      SET , -      : Set up for source or update
0000 77      NUP , -      : No more updates to process
0000 78      SRC , -      : Next line from source file
0000 79      UPD , -      : Next line from update file
0000 80      UPE , -      : Report update errors
0000 81      UPR , -      : Update ready
0000 82      BLK , -      : Process next edit block of update
0000 83      GET , -      : Get next update line
0000 84      EOF >       : End of file
0000 85      :
0000 86      :
0000 87      : Procedure flag byte definitions
0000 88      :
0000 89      -VIELD PRC,0,< -
0000 90      <EXPED,,M> -   : Expected edit command
0000 91      <DELINÉ,,M> -  : Deleted lines information pending
0000 92      <ERRORS,,M> -  : Clash errors to report
0000 93      <HIEDIT,,M> -  : Highest precedence edit overrides others
0000 94      <NODATA,,M> -  : Data from edit being ignored
0000 95      >
0000 96      :
0000 97      :
0000 98      :
0000 99      : Local storage
0000 100     :
0000 101     :
00000000 102     .PSECT  SUM$RW_DATA,NOEXE, LONG
0000 103     :
0000 104     :
00000000 105     SUM_CUR_RAB:      : Address of the currently active RAB.
0000 106     .LONG      0
0004 107     :
00000000 108     SUM_UBF_ADDR:      : Address of local UBF. The size of
0004 109     .LONG      0          : the UBF is established by the size
0008 110     :                  : of the main program's (caller's) RAB.
0008 111     :
00000000 112     .PSECT  SUM$RO_DATA,NOEXE,NOWRT, LONG
```


DECLARATIONS

	0000	113	:		
	0000	114	:		
00000082	0000	115	SUM_ISSZE:		
	0000	116	.LONG	IS_K_BLN	: Size of input stream block
	0004	117	:		
0000001A	0004	118	SUM_EDSZE:		
	0004	119	.LONG	ED_K_BLN	: Size of Edit block

```
TPARSE
0008 121 .SBTTL TPARSE
0008 122 ;
0008 123 .SAVE
0008 124
00000008 125 .PSECT SUM$RW_DATA,NOEXE,LONG
0008 126 ;
0008 127 ;
0008 128 TPARSE_BLOCK:
00000008 0008 129 .LONG TPASK_COUNT0
0000002C 000C 130 .BLKB TPASK_LENGTH0-4
002C 131 ;
002C 132 ; Continue Tparse parameter block with own data
002C 133 ;
002C 134 SUM_TPARSE:
002C 135 ;
00000024 002C 136 TPA_W_LOC1 = .-TPARSE_BLOCK
0000002E 002C 137 .BLKW 1
00000026 002E 138 TPA_W_LOC2 = .-TPARSE_BLOCK
00000030 002E 139 .BLKW 1
00000028 0030 140 TPA_B_ISFLAGS = .-TPARSE_BLOCK
00000031 0030 141 .BLKB 1
00000029 0031 142 TPA_B_EDFLAGS = .-TPARSE_BLOCK
00000032 0031 143 .BLKB 1
0000002A 0032 144 TPA_W_DOT = .-TPARSE_BLOCK
00000034 0032 145 .BLKW 1
0000002C 0034 146 TPA_W_LOC = .-TPARSE_BLOCK
00000036 0034 147 .BLKW 1
0000002E 0036 148 TPA_W_LINTYP = .-TPARSE_BLOCK
00000038 0036 149 .BLKW 1
00000030 0038 150 TPA_Q_AUDDS = .-TPARSE_BLOCK
00000040 0038 151 .BLKQ 1
00000038 0040 152 TPA_Q_CMNT = .-TPARSE_BLOCK
00000048 0040 153 .BLKQ 1
00000040 0048 154 TPA_Q_LINEDS = .-TPARSE_BLOCK
00000050 0048 155 .BLKQ 1
0050 156 ;
0050 157 ;
00000008 158 .PSECT SUM$RO_DATA
0008 159 ;
0000002C 0008 160 COMMA = ^X2C
0000003B 0008 161 SEMICOLON = ^X3B
0000003C 0008 162 LESSTHAN = ^X3C
0008 163 ;
0008 164 $INIT_STATE MER_STATE,MER_KEY
0008 165 ;
0008 166 ; Get 1st character of line
0008 167 ;
0008 168 $STATE
0008 169 $STRAN TPA$_LAMBDA,,ACT_BLANKS_SIG
0008 170 $STATE
0008 171 $STRAN '-' ,EDIT
0008 172 $STRAN '%' ,CMND,ACT_PERCENT
0008 173 $STRAN '/' ,TERM
0008 174 $STRAN LESSTHAN ,DATA,ACT_ESC
0008 175 $STRAN '@' ,TPA$_FAIL
0008 176 $STRAN '\' ,CMND,ACT_BACKSLASH
0008 177 $STRAN TPA$_EOS ,DATA
```

TPARSE

```
0008 178          $STRAN  TPAS_ANY,DATA
0008 179          :
0008 180          : End data line
0008 181          :
0008 182          $STATE  DATA
0008 183          $STRAN  TPAS_LAMBDA,TPAS_EXIT,ACT_EXIT,,,0
0008 184          :
0008 185          : End normal command line
0008 186          :
0008 187          $STATE  CMND
0008 188          $STRAN  TPAS_LAMBDA,TPAS_EXIT,ACT_EXIT,,,CMD_M_CMND
0008 189          :
0008 190          : End data terminating command
0008 191          :
0008 192          $STATE  TERM
0008 193          $STRAN  TPAS_LAMBDA,TPAS_EXIT,ACT_EXIT,,, -
0008 194                      <CMD_M_CMND!CMD_M_EDTRM!CMD_M_EDEND>
0008 195          :
0008 196          :
0008 197          : Edit command
0008 198          :
0008 199          : Read locator-1
0008 200          :
0008 201          $STATE  EDIT
0008 202          $STRAN  '-',ACT_SUPPRESS
0008 203          $STRAN  TPAS_LAMBDA
0008 204          $STATE
0008 205          $STRAN  TPAS_LAMBDA,,ACT_BLANKS_NSIG
0008 206          $STATE
0008 207          $STRAN  !LOCATOR,,ACT_LOC1
0008 208          :
0008 209          : Read Locator-2
0008 210          :
0008 211          $STATE
0008 212          $STRAN  TPAS_EOS,TPAS_EXIT
0008 213          $STRAN  SEMICOLON,CMNT,ACT_CMNT
0008 214          $STRAN  COMMA
0008 215          $STATE
0008 216          $STRAN  !LOCATOR,,ACT_LOC2
0008 217          $STRAN  TPAS_EOS,TPAS_EXIT
0008 218          : Read audit string
0008 219          :
0008 220          $STATE
0008 221          $STRAN  TPAS_EOS,TPAS_EXIT
0008 222          $STRAN  SEMICOLON,CMNT,ACT_CMNT
0008 223          $STRAN  COMMA
0008 224          $STATE
0008 225          $STRAN  '/',ACT_AUDIT
0008 226          $STRAN  TPAS_EOS,TPAS_EXIT
0008 227          $STRAN  SEMICOLON,CMNT,ACT_CMNT
0008 228          $STATE  AUDCH
0008 229          $STRAN  '/',ACT_AUDEND
0008 230          $STRAN  TPAS_ANY,AUDCH,ACT_AUDCH
0008 231          :
0008 232          : Read comment line
0008 233          :
0008 234          $STATE
```


TPARSE

```
0008 235      $STRAN  TPAS_EOS,TPAS_EXIT
0008 236      $STRAN  SEMICOLON,CMNT,ACT_CMNT
0008 237      $STATE  CMNT
0008 238      $STRAN  TPAS_LAMBDA,TPAS_EXIT
0008 239      :
0008 240      :
0008 241      : Subexpression to parse locator
0008 242      :
0008 243      $STATE  LOCATOR
0008 244      $STRAN  ' ',ACT_DOT
0008 245      $STRAN  TPAS_DECIMAL,,ACT_LOCNUM
0008 246      $STRAN  TPAS_LAMBDA,TPAS_EXIT
0008 247      $STATE
0008 248      $STRAN  '+'
0008 249      $STRAN  TPAS_LAMBDA,TPAS_EXIT
0008 250      $STATE
0008 251      $STRAN  TPAS_DECIMAL,,ACT_PLUS
0008 252      $STATE
0008 253      $STRAN  TPAS_LAMBDA,TPAS_EXIT
0008 254      :
0008 255      $END_STATE
0008 256      :
0008 257      :
00000008 258      .RESTORE
```

SUM\$INIT

```
0008 260 .SBTTL SUM$INIT
0008 261 :
0008 262 :++
0008 263 : Functional description:
0008 264 :
0008 265 : This procedure is called to initialise the update files.
0008 266 :
0008 267 :
0008 268 : Input parameters:
0008 269 :
0008 270 : 4(AP) = Address of input stream control block
0008 271 : 8(AP) = Address of update files list
0008 272 : 12(AP) = Address of main program RAB
0008 273 :
0008 274 :
0008 275 : Outputs:
0008 276 :
0008 277 : IS_L_MAIN_FAB(R9) = FAB address of source file
0008 278 :
0008 279 : Implicit outputs:
0008 280 :
0008 281 : The edit nodes list.
0008 282 :
0008 283 : SUM_UBF_ADDR points to the local UBF which is allocated (if it has
0008 284 : not been previously).
0008 285 :
0008 286 :--
0008 287 :
00000000 288 .PSECT SUM$CODE,NOWRT, LONG
0000 289 :
02 OFFC 0000 290 .ENTRY SUM$INIT_EDIT, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
11 0002 291 BRB SUM$INIT
0004 292 :
OFFC 0004 293 .ENTRY SUM$INIT_CMND, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
0006 294 :
0006 295 SUM$INIT:
50 01 D0 0006 296 MOVL #1,R0 ; Assume successful completion
58 04 AC D0 0009 297 MOVL 4(AP),R8 ; Get address of SUM control block
59 04 A8 D0 000D 298 MOVL SUM_L_ISDATA(R8),R9 ; Get input stream data block address
0A 12 0011 299 BNEQ 5$ ; Branch if block has been allocated
00A4 30 0013 300 BSBW GET_IS_BLK ; Get and initialise data block
6D 50 E9 0016 301 BLBC R0,7$ ; Error of LBC
04 AB 59 D0 0019 302 MOVL R9,SUM_L_ISDATA(R8) ; Save data block address
001D 303 5$:
04 18 A8 B4 001D 304 CLRW SUM_W_LINE_NO(R8) ; Reset return line number
06 A9 00 90 0020 305 MOVW #SUM_ST_SET,IS_B_STATE(R9) ; Initialise state to SET
06 A9 01 B0 0024 306 MOVW #1,IS_W_LINE_NO(R9) ; and source file line number
5A 08 AC D0 0028 307 MOVL 8(AP),R10 ; Get file list address
58 0C AC D0 002C 308 MOVL 12(AP),R8 ; Get RAB address
00000004'EF D5 0030 309 TSTL SUM_UBF_ADDR ; Has a UBF been allocated?
19 12 0036 310 BNEQ 6$ ; If NEQ a UBF already exists.
7E 20 A8 3C 0038 311 MOVZWL RAB$W_USZ(R8),-(SP) ; Set up the buffer size.
00000004'EF 9F 003C 312 PUSHAB SUM_UBF_ADDR ; Stack arguments for LIB$GET_VM
04 AE DF 0042 313 PUSHAL 4(SP) ;
00000000'GF 02 FB 0045 314 CALLS #2, G^LIB$GET_VM ; Allocate a local UBF.
37 50 E9 004C 315 BLBC R0,7$ ; Error if LBC
BE D5 004F 316 TSTL (SP)+ ; Clean up the stack.
```

SUM\$INIT									
20	A9	58	D0	0051	317	5\$:	MOVL	R8,IS_L_MAIN_RAB(R9)	: Save RAB address
		10	A8	D4	0055	318	CLRL	RAB\$W-RFA+0(R8)	: Clear RFA
		14	A8	B4	0058	319	CLRW	RAB\$W-RFA+4(R8)	: (3 words)
		04	8B	30	005B	320	BSBW	SAVE_SRC_RFA	: and save it
1C	A9	3C	A8	D0	005E	321	MOVL	RAB\$C_FAB(R8),IS_L_MAIN_FAB(R9)	: Save FAB address
		69	5A	D0	0063	322	MOVL	R10,IS_L_FILELIST(R9)	: Save file list address
			51	13	0066	323	BEQL	40\$: If EQL there is no list so return
40	08	AA	00	E2	0068	324	BBSS	#UPF_V_INIT,-	: Branch if already initialised
					006D	325		UPF_B_FIFLAGS(R10),30\$	
10	AA	10	AA	DE	006D	326	MOVAL	UPF_Q_EDITS(R10),-	: Initialise edit list head in
					0072	327		UPF_Q_EDITS(R10)	: first file block
14	AA	10	AA	DE	0072	328	MOVAL	UPF_Q_EDITS(R10),-	
					0077	329		UPF_Q_EDITS+4(R10)	
					0077	330		\$DISCONNECT_RAB=R8,ERR=SUM\$CLOSE_ERR	: Disconnect RAB
					0086	331			
		30	50	E9	0086	332	BLBC	R0,40\$: Error if LBC
					0089	333			
			64	10	0089	334	BSB	PROCESS_FILE	: Process update files
		05	50	E9	008B	335	BLBC	R0,20\$: Error if LBC
		5A	6A	D0	008E	336	MOVL	(R10),R10	: Get next file block address
			F6	12	0091	337	BNEQ	10\$: End of list if EQL
					0093	338			
		5A	69	D0	0093	339	MOVL	IS_L_FILELIST(R9),R10	: Reset file list pointer
3C	A8	1C	A9	D0	0096	340	MOVL	IS_L_MAIN_FAB(R9),RAB\$C_FAB(R8)	: Reset FAB address
					009B	341	\$CONNECT	RAB=R8,ERR=SUM\$OPEN_ERR	
		04	47	30	00AA	342	BSBW	RESTORE_SRC_RFA	: Restore source file RFA
					00AD	343			
10	A9	10	AA	D0	00AD	344	MOVL	UPF_Q_EDITS(R10),IS_L_EDIT_BLK(R9)	: Reset edit block pointer
	2A	A9	03	88	00B2	345	BISB2	#SUM_M_AUDIT!SUM_M_AUDITNEW,-	: Switch on audit trail and
					00B6	346		IS_B_F[AGS(R9)	: mark first audit as new
		30	A9	B4	00B6	347	CLRW	IS_W_DELETES(R9)	: Initialise number of deleted lines
					00B9	348			
				04	00B9	349	RET		

GET_IS_BLK

.SBTTL GET_IS_BLK

..

++

Functional description:

This routine obtains a memory block for an input stream data
block and if successful initialises the block.

Inputs:

None

Outputs:

R9 = Address of memory block

..

--

GET_IS_BLK:

```
371 PUSHAB SUM$VIRT ADDR      ; Stack arguments for LIB$GET_VM
372 PUSHAB SUM_ISSZE          ;
373 CALLS #2,6*LIB$GET_VM      ; Get memory block
374 BLBC R0,10$                ; Error if LBC
375 MOVL SUM$VIRT_ADDR,R9      ; Get block address
376 MOVCS #0,(R9),#0,#IS_K_BLN,(R9) ; Clear block
377 MOVAB IS_T_FAB(R9),RT      ; Set FAB block pointer
378 $FAB_STORE FAB = R1, -     ; and initialise as a FAB
379 BID = #FAB$C_BID, -
380 BLN = #FAB$C_BLN
381 MOVL #1,R0                  ; Set success status
382 10$:
383 RSB
```

```
00000000'EF 9F 00BA 371
00000000'EF 9F 00C0 372
00000000'GF 02 FB 00C6 373
1E 50 E9 00CD 374
59 00000000'EF D0 00D0 375
69 0082 8F 00 69 00 2C 00D7 376
51 32 A9 9E 00DF 377
00E3 378
00E3 379
00E3 380
50 01 D0 00EB 381
00EE 382
05 00EE 383
```

PROCESS_FILE

```
00EF 385      .SBTTL PROCESS_FILE
00EF 386      :
00EF 387      :++
00EF 388      : Functional description:
00EF 389      :
00EF 390      :     This routine is called to process each update file
00EF 391      :
00EF 392      : Inputs:
00EF 393      :     R8 = RAB address
00EF 394      :     R9 = Input stream data block address
00EF 395      :     R10 = File node address
00EF 396      :
00EF 397      : Outputs:
00EF 398      :
00EF 399      :     R0 = Success/error status
00EF 400      :
00EF 401      : Implicit outputs:
00EF 402      :
00EF 403      :     Edit blocks list
00EF 404      :
00EF 405      :--
00EF 406      : PROCESS_FILE:
5A A9 38 AA 9E 00EF 407      : MOVAB    UPF T_NAM(R10), -      ; Set NAM block pointer
00 36 A9 18 E2 00F4 408      : IS T_FAB+RAB$F_L_NAM(R9)
00F4 409      : BBSS      #FAB$V_NAM, -      ; Set for open by NAM block
00F9 410      : IS T_FAB+RAB$F_L_FOP(R9), 5$
00F9 411      : 5$:
00F9 412      : $OPEN    FAB=IS_T_FAB(R9),ERR=SUM$OPEN_ERR      ; Open input file
4D 50 E9 0109 413      : BLBC      R0,30$      ; Error if LBC
3C A8 32 A9 DE 010C 414      : MOVAL    IS T_FAB(R9),RAB$F_L_FAB(R8)      ; Put FAB address into RAB
0111 415      : $CONNECT  RAB=R8,ERR=SUM$OPEN_ERR      ; Connect RAB to FAB
26 50 E9 0120 416      : BLBC      R0,20$      ; Error if LBC
0123 417      : $FIND     RAB=R8,ERR=SUM$READ_ERR      ; Initialise RFA
05 50 E9 0132 418      : BLBC      R0,10$      ; Error if LBC
0135 419      :
10 A9 D4 0135 420      : CLRL     IS_L_EDIT_BLK(R9)      ; Clear last edit node address
0138 421      :
0138 422      : Read update file and create edit nodes
0138 423      :
20 10 0138 424      : BSBB     SET_UP_NODES      ; Read update file
013A 425      :
013A 426      : 10$:
013A 427      : $DISCONNECT RAB=R8,ERR=SUM$CLOSE_ERR
0149 428      : 20$:
0149 429      : $CLOSE    FAB=IS_T_FAB(R9),ERR=SUM$CLOSE_ERR      ; Close input file
0159 430      : 30$:
05 0159 431      : RSB
```

SET_UP_NODES

```
015A 433 .SBTTL SET_UP_NODES
015A 434
015A 435 Subroutine to form all edit_nodes
015A 436
015A 437 Inputs:
015A 438 R8 = RAB address
015A 439 R10 = file node address
015A 440
015A 441 Outputs:
015A 442 R0 = Success/error status
015A 443
015A 444
015A 445 SET_UP_NODES:
015A 446 ASSUME UPF_W_LOC2 EQ <UPF_W_LOC1+2>
015A 447 ASSUME ED_W_LOC2 EQ <ED_W_LOC1+2>
015A 448 10$:
015A 449 PUSHAB SUMSVIRT_ADDR ; Stack arguments for LIB$GET_VM
0160 450 PUSHAB SUM_EDSIZE ;
0166 451 CALLS #2,G*LIB$GET_VM ; Get edit block
016D 452 BLBC R0,70$ ; Error if LBC
0170 453 MOVL SUMSVIRT_ADDR,R11 ; Set block pointer
0177 454 MOVL R10,ED_W_LOC1(R11) ; Fill in file block address
017B 455 MOVB UPF_B_FILENO(R10), - ; and file number
0180 456 ED_W_FILENO(R11)
0180 457 MOVL RABSW_RFA+0(R8),ED_W_RFA+0(R11) ; Record file address (3 words)
0185 458 MOVW RABSW_RFA+4(R8),ED_W_RFA+4(R11)
018A 459 CLRW ED_W_LINES(R11)
018D 460 MOVL UPF_W_LOC1(R10),ED_W_LOC1(R11) ; Move both locator numbers
0192 461 MOVB UPF_B_EDFLAGS(R10),ED_B_FLAGS(R11) ; and flags to edit node
0197 462 30$:
0197 463 BSBW READ_UPD_LINEA ; Read line from input file
019A 464 BLBS R0,40$ ; OK if LBS
019D 465 CMPL R0,#RMS$_EOF ; Is error end-of-file?
01A4 466 BNEQ 80$ ; No if NEQ
01A6 467 MOVL #CMD_M_ALL,R4 ; Fake an end-of-edit command
01A9 468 BRB 50$ ; Error will be reported on next pass
01AB 469 40$:
01AB 470 BSBW COMMAND_CHECK ; Check for command
01AE 471 BLBC R0,30$ ; Syntax error if LBC
01B1 472 BBS #CMD_V_EDTRM,R4,50$ ; Branch if data terminating command
01B5 473 BBS #CMD_V_CMND,R4,30$ ; Branch if normal command
01B9 474 INCW ED_W_LINES(R11) ; Increment number of insert lines for
01BC 475 BRB 30$ ; this edit
01BE 476 50$:
01BE 477 TSTL ED_W_LOC1(R11) ; If Loc-1 and Loc-2 = 0 and Lines <> 0
01C1 478 BNEQ 60$ ; there is an insert in front of
01C3 479 ; the file, otherwise throw this
01C3 480 ; Edit node away
01C3 481 TSTW ED_W_LINES(R11)
01C6 482 BNEQ 60$
01C8 483 BBC #CMD_V_EDEND,R4,60$ ; Branch if not end of edits
01CC 484 PUSHAB SUMSVIRT_ADDR ; Stack arguments for LIB$FREE_VM
01D2 485 PUSHAB SUM_EDSIZE ;
01D8 486 CALLS #2,G*LIB$FREE_VM ; Return unused memory block
01DF 487 BLBC R0,70$ ; Error if LBC
01E2 488 BRB 80$
01E4 489 60$:
```


SET_UP_NODES

	0F	10	01E4	490	
OA 54	02	E0	01E6	491	
	FF6D	31	01EA	492	
			01ED	493	70\$:
00000000'EF	00	FB	01ED	494	
			01F4	495	80\$:
		05	01F4	496	

BSB
BBS
BRW

CALLS

RSB

INSERT_NODE
#CMD_V_EDEND,R4,80\$
10\$

#0,SUM\$LIB_ERR

; Insert block into edits list
; Branch if edit terminating command
; Go back for next edit command

; Report error

N 1

16-SEP-1984 02:10:14 VAX/VMS Macro V04-00
5-SEP-1984 03:38:52 [SUM.SRC]SUMEDIT.MAR;1

INSERT_NODE

```
01F5 498      .SBTTL INSERT_NODE
01F5 499
01F5 500      Subroutine to insert block into edit list
01F5 501
01F5 502      This routine checks that the edit node is in sequence with any other nodes
01F5 503      from the same update file. If not, the edit node is marked so that a
01F5 504      warning can be produced later. However, the node is placed in the correct
01F5 505      position.
01F5 506
01F5 507      Inputs:
01F5 508          R11 = address of block to insert
01F5 509          IS_L_EDIT_BLK(R9) = Last edit node inserted from current update file
01F5 510
01F5 511      Outputs:
01F5 512          None
01F5 513
01F5 514
01F5 515      INSERT_NODE:
50 08 AC D0 01F5 516      MOVL      8(AP),R0          ; Get address of first file block
50 10 A0 DE 01F9 517      MOVAL     UPF_Q_EDITS(R0),R0      ; and form edit list head address
51 10 A9 D0 01FD 518      MOVL      IS_L_EDIT_BLK(R9),R1      ; Get address of last node inserted
                    05 12 0201 519      BNEQ      10$          ; If NEQ there is one
                    51 50 D0 0203 520      MOVL      R0,R1          ; This is first node so scan list
                    10 11 0206 521      BRB       20$          ; from list head
                    0208 522
10$ 08 A1 08 AB B1 0208 523      CMPW      ED_W_LOC1(R11),ED_W_LOC1(R1) ; Is edit out of sequence?
                    09 14 020D 524      BGTR      20$          ; No if GTR
18 AB 02 88 020F 525      BISB      #ED_M_SEQERR,ED_B_FLAGS(R11) ; Mark edit node
                    51 50 D0 0213 526      MOVL      R0,R1          ; Scan list from list head to find
                    04 11 0216 527      BRB       30$          ; correct position
                    0218 528
20$ 10 A9 5B D0 0218 529      MOVL      R11,IS_L_EDIT_BLK(R9) ; Set new 'last edit' address
                    021C 530
30$ 51 61 D0 021C 531      MOVL      (R1),R1          ; Get next block
                    50 51 D1 021F 532      CMPL      R1,R0          ; At end of list?
                    07 13 0222 533      BEQL      40$          ; Yes if EQL
08 A1 08 AB B1 0224 534      CMPW      ED_W_LOC1(R11),ED_W_LOC1(R1) ; Is new LOC-1 <= current LOC-1
                    F1 14 0229 535      BGTR      30$          ; No if GTR
                    022B 536
40$ 04 B1 5B 0E 022B 537      INSQUE     (R11),#ED_L_BWD(R1) ; Insert new node into list
                    05 022F 538      RSB
```

READ_UPD_LINE

```
0230 540      .SBTTL READ_UPD_LINE
0230 541      :
0230 542      Subroutine to read line sequentially from current update file
0230 543      :
0230 544      There are two entry points:
0230 545      :
0230 546      READ_UPD_LINE to access the file and read line
0230 547      :
0230 548      READ_UPD_LINEA if update file is already accessed and ready
0230 549      for next line to be read
0230 550      :
0230 551      :
0230 552      Inputs:
0230 553      R8 = RAB address for reading file
0230 554      :
0230 555      Implicit Inputs:
0230 556      SUM_UBF_ADDR address of local UBF, to avoid access conflicts.
0230 557      :
0230 558      Outputs:
0230 559      R0 = success/error status
0230 560      R6 = Line size
0230 561      R7 = Line buffer address
0230 562      :
0230 563      .ENABL LSB
0230 564      :
0230 565      :
0230 566      READ_UPD_LINE:
0230 567      BSBW ACCESS_UPDATE      : Access update file
0230 568      BLBC R0,10$            : Error if LBC
0230 569      :
0230 570      READ_UPD_LINEA:
0230 571      PUSHL RAB$L_UBF(R8)      : Save the old UBF address.
0230 572      PUSHL RAB$L_ROP(R8)      : Save the old ROP field.
0230 573      BICL2 #RAB$M_LOC, RAB$L_ROP(R8) : Set MOVE mode for $GET.
0230 574      MOVL SUM_UBF_ADDR, RAB$L_UBF(R8) : Use local buffer.
0230 575      $GET RAB = R8, ERR = SUM$READ_ERR : Read line
0230 576      MOVL (SP)+, RAB$L_ROP(R8) : Restore old ROP
0230 577      MOVL (SP)+, RAB$L_UBF(R8) : and UBF.
0230 578      BLBC R0,10$            : If error, don't copy string.
0230 579      MOVZWL RAB$W_RSZ(R8),R6 : Set line size
0230 580      MOVL RAB$L_RBF(R8),R7 : and buffer address
0230 581      BISB2 #SUM_M_SRCUPD,IS_B_FLAGS(R9) : Mark as update line
0230 582      BBS #RAB$V_LOC, RAB$E_ROP(R8), 10$ : Should we copy string to UBF?
0230 583      PUSHF #^M<R0,R1,R2,R3,R4,R5> : Save registers across MOVCL3
0230 584      MOVCL3 RAB$W_RSZ(R8),- : String length
0230 585      @SUM_UBF_ADDR,- : Source buffer
0230 586      @RAB$L_UBF(R8) : Destination buffer
0230 587      POPF #^M<R0,R1,R2,R3,R4,R5> : Restore registers
0230 588      RSB
0230 589      :
0230 590      .DSABL LSB
```

030D 30 0230 567
4F 50 E9 0233 568
24 A8 DD 0236 569
04 A8 DD 0239 571
04 A8 DD 023C 572
24 A8 DD 0244 573
04 A8 DD 024C 574
24 A8 DD 025B 575
1F 50 DD 025F 576
56 22 A8 DD 0263 577
57 28 A8 DD 0266 578
2A A9 04 DD 026A 579
OE 04 A8 DD 026E 580
22 A8 DD 0272 581
00000004'FF DD 0277 582
24 B8 DD 0279 583
3F BA DD 0281 584
05 05 DD 0283 585
0286 586
0288 587
0289 588
028A 589
028B 590


```
SUM$LINE
0286 592 .SBTTL SUM$LINE
0286 593
0286 594 : This procedure is called from the main program to get the next
0286 595 input line. This line may come from either the source file or
0286 596 an update file.
0286 597
0286 598 Inputs:
0286 599
0286 600 4(AP) = Address of control block
0286 601
0286 602 Outputs:
0286 603
0286 604 Next line
0286 605
0286 606
0286 607 .ENTRY SUM$LINE,*M(R2,R3,R4,R5,R6,R7,R8,R9,R10,R11)
0288 608 MOVL 4(AP),R1 ; Get address of control block
028C 609 MOVL SUM_L_ISDATA(R1),R9 ; Set input stream data block address
0290 610 MOVL IS_L_MAIN_RAB(R9),R8 ; Get main program RAB address
0294 611 MOVL IS_L_EDIT_BLK(R9),R11 ; Get current edit block address
0298 612 SUM_DISPATCH:
0298 613 CASEB IS_B_STATE(R9),#SUM_ST_SET,#SUM_ST_EOF ; Branch to service routine
029D 614 10$: .SIGNED_WORD LINE_SET-10$
029F 615 .SIGNED_WORD LINE_NUP-10$
02A1 616 .SIGNED_WORD LINE_SRC-10$
02A3 617 .SIGNED_WORD LINE_UPD-10$
02A5 618 .SIGNED_WORD LINE_UPE-10$
02A7 619 .SIGNED_WORD LINE_UPR-10$
02A9 620 .SIGNED_WORD LINE_BLK-10$
02AB 621 .SIGNED_WORD LINE_GET-10$
02AD 622 .SIGNED_WORD LINE_EOF-10$
02AF 623
02AF 624 SUM_RETURN:
02AF 625 MOVL R11,IS_L_EDIT_BLK(R9) ; Preserve edit block address
02B3 626 MOVL 4(AP),R1 ; Get address of control block
02B7 627 MOVL R0,SUM_L_STS(R1) ; Return status
02BA 628 MOVB IS_B_FLAGS(R9),- ; Edit flags
02BF 629 SUM_B_FLAGS(R1)
02C4 630 BBC #SUM_V_SRCUPD,- ; Branch if source line
02C4 631 SUM_B_FLAGS(R1),5$
02C4 632 MOVW IS_Q_INSERT_NO(R9),SUM_W_INSERT_NO(R1) ; Inserts
02C9 633 MOVQ UPF_Q_AUDDS(R10),- ; Supply audit string descriptor
02CE 634 SUM_Q_AUDDS(R1)
02CE 635 MOVAL UPF_T_NAM(R10),R10 ; Form NAM block address
02D2 636 MOVZBL NAM$B_RSL(R10),- ; Get file spec size
02D7 637 SUM_Q_FILESP+0(R1)
02D7 638 MOVL NAM$L_RSA(R10),- ; and address
02DC 639 SUM_Q_FILESP+4(R1)
02DC 640 BLBS R0,T0$ ; If error line
02DF 641 BBSC #SUM_V_SRCUPD,SUM_B_FLAGS(R1),10$ ; don't mark as update line
02E4 642
02E4 643 : Source file line
02E4 644
02E4 645 1$:
02E4 646 SUBW3 #1,IS_W_LINE_NO(R9),SUM_W_LINE_NO(R1) ; Number of line being returned
02EA 647 BLBC R0,10$ ; If error save deleted line information
02ED 648 ; until first good line
```

08 00 04 A9 8F 0298 612
51 04 AC D0 0288 608
59 04 A1 D0 028C 609
58 20 A9 D0 0290 610
5B 10 A9 D0 0294 611
0063' 029D 614
008A' 029F 615
0090' 02A1 616
00A1' 02A3 617
0127' 02A5 618
014F' 02A7 619
016E' 02A9 620
0191' 02AB 621
0204' 02AD 622
02AF 623
02AF 624
10 A9 5B D0 02AF 625
51 04 AC D0 02B3 626
61 50 D0 02B7 627
1C A1 2A A9 90 02BA 628
02BF 629
20 1C A1 02 E1 02BF 630
02C4 631
1A A1 2E A9 B0 02C4 632
08 A1 18 AA 7D 02C9 633
02CE 634
5A 38 AA DE 02CE 635
10 A1 03 AA 9A 02D2 636
02D7 637
14 A1 04 AA D0 02D7 638
02DC 639
1B 1C A1 20 50 E8 02DC 640
02DF 641
02E4 642
02E4 643
02E4 644
02E4 645
18 A1 06 A9 01 A3 02E4 646
12 50 E9 02EA 647
02ED 648

				SUM\$LINE			
0D	05	A9	01	E5	02ED	649	BBCC
					02F2	650	
1A	A1	30	A9	B0	02F2	651	MOVW
03	1C	A1	04	E2	02F7	652	BBSS
					02FC	653	
		30	A9	B4	02FC	654	CLRW
					02FF	655	
				04	02FF	656	RET

10\$:

#PRC_V DELINE - ; Branch if no pending deleted info
IS_B-PROCFAGS(R9),10\$
IS-W-DELETES(R9),SUM_W_INSERT_NO(R1) ; Return number of lines delete
#SDM-V DELETE - ; Set deleted lines information flag
SUM_B_FLAGS(R1),10\$
IS-W-DELETES(R9) ; Reset number of deleted lines

```
LINE_SET
0300 658 .SBTTL LINE_SET
0300 659
0300 660 Routine to service SET state
0300 661 Determines if the next line is to come from the main source file
0300 662 or from an update file. If there are no more updates to be processed
0300 663 the state is set to NUP; if there are updates but the next update is to
0300 664 be applied to a later source line the state is set to SRC; if the next
0300 665 line is to come from an update file the state is set to UPD.
0300 666
0300 667
0300 668 Inputs:
0300 669
0300 670 R11 = Current edit block address
0300 671
0300 672 Outputs:
0300 673
0300 674 state changed
0300 675
0300 676 LINE_SET:
04 A9 01 90 0300 677 MOVB #SUM_ST_NUP,IS_B_STATE(R9) ; Assume no more updates
51 51 69 D0 0304 678 MOVL IS_L_FILELIST(R9),R1 ; Get address of first file block
18 13 0307 679 BEQL 10$ ; If EQL there are no update files
51 10 A1 DE 0309 680 MOVAL UPF_Q_EDITS(R1),R1 ; Form edit block list head address
SB 51 D1 030D 681 CMPL R1,R1 ; Any edits still in list?
OF 13 0310 682 BEQL 10$ ; No if EQL so must be source line
04 A9 03 90 0312 683 MOVB #SUM_ST_UPD,IS_B_STATE(R9) ; Assume next line is from update file
08 AB 06 A9 B1 0316 684 CMPW IS_W_LINE_NO(R9), - ; Is line number of source file less
031B 685 ED_W_LOC1(R11) ; than locator-1 of next edit?
031B 686 BGEQ 20$ ; No if GEQ
04 A9 02 90 031D 687 MOVB #SUM_ST_SRC,IS_B_STATE(R9) ; Change state to source
0321 688 10$:
0187 30 0321 689 BSDW ACCESS_SRC ; Access source file
0324 690 20$:
FF71 31 0324 691 BRW SUM_DISPATCH ; and dispatch again
```



```
LINE_NUP
0327 693 .SBTTL LINE_NUP
0327 694 :
0327 695 : There are no more updates to process so just read next source line
0327 696 : and return to caller. The source file is already accessed so
0327 697 : READ_SRC_LINEA can be used.
0327 698 :
0327 699 :
0327 700 : Inputs:
0327 701 :
0327 702 : None
0327 703 :
0327 704 : Outputs:
0327 705 :
0327 706 : None
0327 707 :
0327 708 :
0327 709 LINE_NUP:
02CA 30 0327 710 BSBW READ_SRC_LINEA : Get next source line
FFB2 31 0327 711 BRW SUM_RETURN : and return
```

LINE_SRC

```
0320 713 .SBTTL LINE_SRC
0320 714 :
0320 715 : The next source line is read from the main input file. The line
0320 716 : number is incremented and compared with the locator-1 value of the
0320 717 : next edit. If the line number remains lower the state remains at SRC.
0320 718 : If the line number is equal or greater the state is changed to UPD.
0320 719 : The next call to SUM$LINE will then get an update line.
0320 720 :
0320 721 : Inputs:
0320 722 :
0320 723 : R11 = Current edit block address
0320 724 :
0320 725 : Outputs:
0320 726 :
0320 727 : state
0320 728 :
0320 729 :
0320 730 LINE_SRC:
0320 731 BSBW READ_SRC LINEA ; Get next line from source file
08 AB 02C4 30 0320 731 BSBW READ_SRC LINEA ; Get next line from source file
08 AB 06 A9 B1 0330 732 CMPW IS_W_LINE NO(R9), - ; Is source line number still lower
0335 733 ED_W_LOC1(R11) ; than next locator-1
04 A9 04 19 0335 734 BLSS 10$ ; Yes if LSS
0337 735 MOVW #SUM_ST_UPD,IS_B_STATE(R9) ; Reset state to UPD
FF71 31 033B 736 10$:
033B 737 BRW SUM_RETURN ; and return with line
```

LINE_UPD

```
033E 739 .SBTTL LINE_UPD
033E 740 :
033E 741 : The next update operation is prepared by determining the range of
033E 742 : the edit, that is the number of edit operations which have clashed.
033E 743 :
033E 744 : Inputs:
033E 745 :
033E 746 : R9 = Input stream data pointer
033E 747 : R11 = Current edit block address
033E 748 :
033E 749 : Outputs:
033E 750 :
033E 751 : IS_L_FIRST_EDIT(R9) = First edit block of update
033E 752 : IS_L_LAST_EDIT(R9) = Last edit block of update
033E 753 : IS_W_HIGH_LOC2(R9) = Highest loc-2 value of update
033E 754 :
033E 755 :
033E 756 LINE_UPD:
033E 757 MOVL R11,IS_L_FIRST_EDIT(R9) ; Save address of first edit
033E 758 MOVZWL ED_W_LOC2(R11),R4 ; Set highest loc-2 value
033E 759 MOVW R4,IS_W_HIGH_LOC2(R9) ; and supply as routine output
033E 760 BICB2 #SUM_M_SUBCLSH,IS_B_FLAGS(R9) ; May be first edit in clash
033E 761 BICB2 #<PRC_M_ERRORS! - ; Assume no clash errors,
033E 762 PRC_M_HIEDIT! - ; highest edit does not override others,
033E 763 PRC_M_NODATA> - ; and all data lines inserted
033E 764 IS_B_PROCFRAGS(R9)
033E 765 MOVW #SOM-ST_UPR,IS_B_STATE(R9)
033E 766 MOVL IS_L_FICELIST(R9),R5 ; Set files list
033E 767 MOVAL UPF_B_EDITS(R5),R5 ; List head address
033E 768 10$: MOVL (R11),R2 ; Point to next edit block
033E 769 CMPL R2,R5 ; At end of list?
033E 770 BEQL 40$ ; Yes if EQL
033E 771 MOVL R4,R1 ; Set highest locator value of edit
033E 772 BNEQ 20$ ; If zero set from loc-2 of current edit
033E 773 MOVZWL ED_W_LOC2(R11),R1 ; Set highest locator value of edit
033E 774 BNEQ 20$ ; If zero set from loc-1 of current edit
033E 775 MOVZWL ED_W_LOC1(R11),R1 ; Set highest locator value of edit
033E 776 20$: CMPW R1,ED_W_LOC1(R2) ; Does this edit overlap with next?
033E 777 BLSS 40$ ; No if LSS
033E 778 :
033E 779 : This edit block clashes with next
033E 780 :
033E 781 :
033E 782 :
033E 783 CMPW R4,ED_W_LOC2(R2) ; Is its loc-2 higher than current loc-2
033E 784 BGEQ 25$ ; No if GEQ
033E 785 MOVZWL ED_W_LOC2(R2),R4 ; Extend range of edit
033E 786 25$: TSTW ED_W_LOC2(R2) ; Is edit all inserts?
033E 787 BEQL 30$ ; Yes if EQL
033E 788 BISB #PRC_M_HIEDIT,IS_B_PROCFRAGS(R9) ; Highest edit overrides others
033E 789 ; therefore replace (later)
033E 790 30$: BSBB CHECK_ERR ; See if error should be reported
033E 791 MOVL R2,R1T ; Point to next edit block
033E 792 BRB 10$
033E 793 40$: MOVL R11,IS_L_LAST_EDIT(R9) ; Set address of last edit block
033E 794
033E 795
```

14	A9	5B	DO	033E	757	
54	0A	AB	3C	0342	758	
2C	A9	54	B0	0346	759	
2A	A9	08	8A	034A	760	
			8A	034E	761	
				034F	762	
05	A9	1C		034F	763	
04	A9	05	90	0352	764	
55	55	69	DO	0356	765	
55	10	A5	DE	0359	766	
				035D	767	
52	6B	DO	035D	768	10\$:	
55	52	D1	0360	769		
	2F	13	0363	770		
51	54	DO	0365	771		
	0A	12	0368	772		
51	0A	AB	3C	036A	773	
	04	12	036E	774		
51	08	AB	3C	0370	775	
				0374	776	
08	A2	51	B1	0374	777	20\$:
	1A	19	0378	778		
			037A	779		
			037A	780		
			037A	781		
0A	A2	54	B1	037A	782	
	04	18	037E	783		
54	0A	A2	3C	0380	784	
			0384	785	25\$:	
	0A	A2	B5	0384	786	
	04	13	0387	787		
05	A9	08	88	0389	788	
			038D	789		
	21	10	038D	790	30\$:	
	5B	52	DO	038F	791	
	C9	11	0392	792		
			0394	793		
18	A9	5B	DO	0394	794	40\$:
				0394	795	


```
LINE_UPD
14 A9 5B D1 0398 796 CMPL R11,IS_L_FIRST_EDIT(R9) ; If first block then single non-clashing
OF 13 039C 797 BEQL 50$ ; edit else last block of clashing edits
10 10 039E 798 BSBB CHECK_ERR ; See if error should be reported
08 05 A9 02 E1 03A0 799 BBC #PRC_V_ERRORS, - ; Branch if no errors to report
03A5 800 IS_B_PROCFLAGS(R9),50$
04 A9 04 90 03A5 801 MOVB #SOM_ST_UPE,IS_B_STATE(R9) ; Set state to report errors
5B 14 A9 D0 03A9 802 MOVL IS_L_FIRST_EDIT(R9),R11 ; Reset edit block pointer to first
FEE8 31 03AD 803 50$: BRW SUM_DISPATCH
03AD 804
03B0 805
03B0 806
03B0 807
03B0 808 : Local subroutine to check if clashing edit should be reported
03B0 809 :
03B0 810 : Inputs:
03B0 811 :
03B0 812 : R11 = Edit block address
03B0 813 :
03B0 814 : Outputs:
03B0 815 :
03B0 816 : None
03B0 817 :
03B0 818 CHECK_ERR:
0E 18 AB 00 E0 03B0 819 BBS #ED_V_SUPPRESS, - ; Branch if suppress bit set
03B5 820 ED_B_FLAGS(R11),20$
08 AB D5 03B5 821 TSTL ED_W_LOC1(R11) ; If Loc-1, Loc-2 and Lines = 0
05 12 03B8 822 BNEQ 10$ ; then do not report as error
0C AB B5 03BA 823 TSTW ED_W_LINES(R11)
04 13 03BD 824 BEQL 20$
03BF 825 10$:
05 A9 04 88 03BF 826 BISB #PRC_M_ERRORS,IS_B_PROCFLAGS(R9) ; Set error report bit
03C3 827 20$:
05 03C3 828 RSB
```

```
LINE_UPE
03C4 830 .SBTTL LINE_UPE
03C4 831 :
03C4 832 : The update operation contains clashing edits which must be reported
03C4 833 :
03C4 834 : Inputs:
03C4 835 :
03C4 836 : R11 = Address of next clashing edit
03C4 837 :
03C4 838 : Outputs:
03C4 839 :
03C4 840 : R11 = Edit block pointer advanced
03C4 841 :
03C4 842 :
03C4 843 LINE_UPE:
50 5A 14 AB D0 03C4 844 MOVL ED_L_FILE(R11),R10 ; Get file block address of clashing edit
FE65 30 03C8 845 BSBW READ_UPD_LINE ; Read update file to get edit line
00848800 8F D0 03CB 846 MOVL #SUMS_EDITSCLSH,R0 ; Set return status
14 A9 5B D1 03D2 847 CMPL R11,IS_L_FIRST_EDIT(R9) ; First report of this set of clashes
04 13 03D6 848 BEQL 10$ ; Yes if EQL
2A A9 08 88 03D8 849 BISB #SUM_M_SUBCLSH,IS_B_FLAGS(R9) ; Set 2nd or later flag
03DC 850 10$:
18 A9 5B D1 03DC 851 CMPL R11,IS_L_LAST_EDIT(R9) ; At last edit?
04 12 03E0 852 BNEQ 20$ ; No if NEQ
04 A9 05 90 03E2 853 MOVB #SUM_ST_UPR,IS_B_STATE(R9) ; Set state to Update Ready
5B 6B D0 03E6 854 20$:
FEC3 31 03E9 855 MOVL (R11),R11 ; Advance to next edit block
856 BRW SUM_RETURN
```

```
LINE_UPR
03EC 858      .SBTTL LINE_UPR
03EC 859
03EC 860      The next update operation is ready. Any errors have been reported
03EC 861      to the caller.
03EC 862
03EC 863
03EC 864      Inputs:
03EC 865
03EC 866      R11 = Current edit block address
03EC 867
03EC 868      Outputs:
03EC 869
03EC 870      None
03EC 871
03EC 872
03EC 873 LINE_UPR:
5B 14 A9 D0 03EC 874      MOVL IS_L_FIRST_EDIT(R9),R11 ; Reset pointer to first edit block
04 A9 06 90 03F0 875      MOVBL #SUM_ST_BLK,IS_B_STATE(R9) ; Reset state to BLK
54 2C A9 3C 03F4 876      MOVZWL IS_W_HIGH_LOC2(R9),R4 ; Is edit operation an insert?
      0B 12 03F8 877      BNEQ 50$ ; No if NEQ
      0B AB B5 03FA 878      TSTW ED_W_LOC1(R11) ; Is insert to front of file?
      09 13 03FD 879      BEQL 60$ ; Yes if EQL
      01E9 30 03FF 880      BSBW READ_SRC_LINE ; Read one more line from source
      FEAA 31 0402 881      BRW SUM_RETURN
      0220 30 0405 882 50$: BSBW SKIP_SRC_LINES ; Skip over source lines to be deleted
      FE8D 31 0408 883 60$: BRW SUM_DISPATCH ; and dispatch
      0408 884
      0408 885
```

```
LINE_BLK
040B 887 .SBTTL LINE_BLK
040B 888 :
040B 889 : This routine is called to begin processing of the next edit block
040B 890 : The file from which edit lines will come is prepared for access. The
040B 891 : state is reset to GET.
040B 892 :
040B 893 :
040B 894 : Inputs:
040B 895 :
040B 896 : R11 = Current edit block address
040B 897 :
040B 898 : Outputs:
040B 899 :
040B 900 : None
040B 901 :
040B 902 :
040B 903 LINE_BLK:
040B 904 MOVL ED_L_FILE(R11),R10 ; Get file block address of file
040B 905 BSBW ACCESS_UPDATE ; Prepare for reading file
040B 906 BLBC R0,20$ ; Error if LBC
040B 907 BBCC #PRC_V_EXPED,IS_B_PROCFLAGS(R9),5$ ; Clear expected edit flag
040B 908 5$:
040B 909 TSTW ED_W_LOC1(R11) ; Is this insert in front of file?
040B 910 BNEQ 10$ ; No if NEQ
040B 911 BBSS #PRC_V_EXPED,IS_B_PROCFLAGS(R9),10$ ; Set expected edit flag
040B 912 10$:
040B 913 MOVB #SUM_ST_GET,IS_B_STATE(R9) ; Reset state to GET
040B 914 BRW SUM_DISPATCH ; and dispatch again
040B 915 20$:
040B 916 BRW SUM_RETURN ; Return to caller with error
```

SA 14 AB D0 040B 904
012E 30 040F 905
16 50 E9 0412 906
00 05 A9 00 E5 0415 907
08 AB B5 041A 908
05 12 041D 909
00 05 A9 00 E2 041F 910
04 A9 07 90 0424 913
FE6D 31 0428 914
FE81 31 042B 915
042B 916


```
LINE_GET
042E 918 .SBTTL LINE_GET
042F 919
0430 920 Routine to get next line from update file
0431 921
0432 922
0433 923 Inputs:
0434 924
0435 925 R11 = Current edit block address
0436 926
0437 927 Outputs:
0438 928
0439 929 R11 = Next edit block address
0440 930
0441 931
0442 932 LINE_GET:
0443 933 MOVL ED_L_FILE(R11),R10 ; Set file block pointer
0444 934 10$:
0445 935 BSBW READ_UPD_LINEA ; Get next line from update file
0446 936 BLBS R0,20$ ; OK if LBS
0447 937 CMPL R0,#RMSS_EOF ; Is error end-of-file?
0448 938 BNEQ 35$ ; No if NEQ
0449 939 MOVL #SUMS_PRMEOF,R0 ; Set premature end-of-file status
0450 940 BRB 40$
0451 941 20$:
0452 942 BSBW COMMAND_CHECK ; Check for syntax and type
0453 943 BLBC R0,80$ ; Syntax error if LBC
0454 944 BBS #CMD_V_CMND,R4,30$ ; Branch if command line
0455 945 BBS #PRC_V_NODATA,- ; Ignore data line if higher precedence
0456 946 IS_B_PROCFLAGS(R9),10$ ; edit is overriding others
0457 947 BRB 90$ ; Return to caller with line
0458 948 30$:
0459 949 BBC #CMD_V_EDTRM,R4,10$ ; Branch if not edit terminating command
0460 950 BBSS #PRC_V_EXPED,IS_B_PROCFLAGS(R9),40$ ; If expecting edit get next lin
0461 951 BBC #ED_V_SEQERR,ED_B_FLAGS(R11),10$ ; Was edit out of sequence?
0462 952 MOVL #SUMS_EDOUTSEQ,R0 ; Yes: report error now
0463 953 35$:
0464 954 BRB 100$
0465 955
0466 956 ; Found end of this set of lines
0467 957
0468 958 40$:
0469 959 CMPL R11,IS_L_LAST_EDIT(R9) ; Last edit block in range?
0470 960 BEQL 60$ ; Yes if EQL
0471 961 BBC #PRC_V_HIEDIT,- ; Branch if concatenating inserts
0472 962 IS_B_PROCFLAGS(R9),50$
0473 963 BISB #PRC_M_NODATA,IS_B_PROCFLAGS(R9) ; Ignore data from other edits
0474 964 50$:
0475 965 MOVB #SUM_ST_BLK,IS_B_STATE(R9) ; Reset state to BLK
0476 966 BRB 70$
0477 967 60$:
0478 968 MOVB #SUM_ST_SET,IS_B_STATE(R9) ; Reset state to SET
0479 969 70$:
0480 970 MOVL (R11),R11 ; Point to next edit block
0481 971 BLBC R0,100$ ; If error return to caller first
0482 972 BRW SUM_DISPATCH ; or dispatch again
0483 973 80$:
0484 974 MOVL #SUMS_SLPSYNERR,R0 ; Set SLP syntax error status
0485 975
0486 976
0487 977
0488 978
0489 979
0490 980
0491 981
0492 982
0493 983
0494 984
```

5A 14 AB D0
FE01 30
12 50 E8
0001827A 8F 50 D1
2F 12
50 00848810 8F D0
28 11
020C 30
44 50 E9
07 54 00 E0
D9 05 A9 04 E0
40 11
D3 54 01 E1
0E 05 A9 00 E2
C9 18 AB 01 E1
50 00848818 8F D0
2C 11
18 A9 5B D1
OF 13
04 05 A9 03 E1
05 A9 10 88
04 A9 06 90
04 11
04 A9 00 90
5B 6B D0
OD 50 E9
FE04 31
50 00848808 8F D0

		LINE_GET					
2E A9	B6	049B	975	90\$:			
		049B	976		INCW	IS_W_INSERT_NO(R9)	; Increment number of new/replace lines
		049E	977	100\$:			
FE0E	31	049E	978		BRW	SUM_RETURN	; Return to caller

LINE_EOF

```
04A1 980      .SBTTL LINE_EOF
04A1 981      :
04A1 982      Routine to service EOF state.  An RMS end-of-file state is
04A1 983      returned to the caller
04A1 984      :
04A1 985      :
04A1 986      Inputs:
04A1 987      :
04A1 988      None
04A1 989      :
04A1 990      :
04A1 991      Outputs:
04A1 992      :
04A1 993      None
04A1 994      :
04A1 995      :
04A1 996      LINE_EOF:
04A1 997      MOVL    #RMSS_EOF,R0          ; Set R0 to eof state
04A8 998      BRW     SUM_RETURN          ; and return to caller
```

50 0001827A 8F D0
FE04 31 04A8

ACCESS_SRC

```
04AB 1000 .SBTTL ACCESS_SRC
04AB 1001
04AB 1002 Routine to access main source file. The RAB is connected to
04AB 1003 the main file FAB if it is not already connected.
04AB 1004
04AB 1005 Inputs:
04AB 1006
04AB 1007 R8 = Main program RAB address
04AB 1008
04AB 1009
04AB 1010 Outputs:
04AB 1011
04AB 1012 None
04AB 1013
04AB 1014
04AB 1015 ACCESS_SRC:
04AB 1016 TSTW RAB$W_ISI(R8) ; Is it connected to a FAB?
04AE 1017 BEQL 10$ ; No if EQL
51 32 A9 DE 04B0 1018 MOVAL IS T FAB(R9),R1 ; Set input stream FAB address
51 3C A8 D1 04B4 1019 CMPL RAB$C_FAB(R8),R1 ; Is it connected to SUM FAB?
2E 12 04B8 1020 BNEQ 20$ ; No if NEQ, it's connected to main FAB
04BA 1021 $DISCONNECT RAB = R8, - ; Disconnect RAB from SUM FAB
04BA 1022 ERR = SUM$CLOSE_ERR
0C A9 D4 04C9 1023 CLRL IS L CONN_FILE(R9) ; Clear file connected flag
19 50 E9 04CC 1024 BLBC R0,20$ ; Error if LBC
04CF 1025 10$:
3C A8 1C A9 D0 04CF 1026 MOVL IS L MAIN_FAB(R9), - ; Put main program FAB into RAB
04D4 1027 RAB$C_FAB(R8)
04D4 1028 $CONNECT RAB = R8, - ; Connect main program FAB to RAB
04D4 1029 ERR = SUM$OPEN_ERR
02 50 E9 04E3 1030 BLBC R0,20$ ; Error if LBC
0C 10 04E6 1031 BSB RESTORE_SRC_RFA ; Restore source file RFA
04E8 1032 20$:
05 04E8 1033 RSB
```


			04E9	1035	.SBTTL	SAVE_SRC_RFA	
			04E9	1036	:		
			04E9	1037	:		
			04E9	1038	:	Routine to save source file record file address	
			04E9	1039	:		
			04E9	1040	:	Inputs:	
			04E9	1041	:		
			04E9	1042	:	R8 = RAB address	
			04E9	1043	:		
			04E9	1044	:	Outputs:	
			04E9	1045	:		
			04E9	1046	:	None	
			04E9	1047	:		
			04E9	1048	:		
			04E9	1049	:	SAVE_SRC_RFA:	
24	A9	10	A8	D0	04E9	1050	MOVL RAB\$W_RFA+0(R8), - ; Move RFA to save buffer
					04EE	1051	IS_W_MAIN_RFA+0(R9)
28	A9	14	A8	B0	04EE	1052	MOVW RAB\$D_RFA+4(R8), -
					04F3	1053	IS_W_MAIN_RFA+4(R9)
		05			04F3	1054	RSB

[illegible]

RESTORE_SRC_RFA

```
04F4 1056 .SBTTL RESTORE_SRC_RFA
04F4 1057
04F4 1058
04F4 1059 Routine to restore source file record file address and
04F4 1060 reset record pointers. If RFA is zero a rewind is performed,
04F4 1061 if non-zero the record is located by a find.
04F4 1062
04F4 1063
04F4 1064 Inputs:
04F4 1065
04F4 1066 R8 = RAB address
04F4 1067
04F4 1068
04F4 1069 Outputs:
04F4 1070
04F4 1071 R0 = Success/error status
04F4 1072
04F4 1073
04F4 1074 RESTORE_SRC_RFA:
10 AB 24 A9 D0 04F4 1075 -MOVE IS W MAIN_RFA+0(R9), - ; Move RFA back to RAB
14 AB 28 A9 B0 04F9 1076 RAB$W_RFA+0(R8) ; (3 words)
04FE 1077 -MOVW IS W MAIN_RFA+4(R9), -
04FE 1078 RAB$W_RFA+4(R8)
04FE 1079 BNEQ 10$ ; If NEQ then do find
10 AB 16 12 0500 1080 TSTL RAB$W_RFA+0(R8) ; Test other part of RFA
11 12 0503 1081 BNEQ 10$ ; If NEQ then do find
0505 1082 $REWIND RAB = R8, - ; Rewind to start of file
0505 1083 ERR = SUM$READ_ERR
29 11 0514 1084 BRB 20$
1E AB 02 90 0516 1085 10$:
051A 1086 -MOVB #RAB$C_RFA,RAB$B_RAC(R8); Put into RFA access mode
051A 1087 $FIND RAB = R8, - ; Reset record pointers
ERR = SUM$READ_ERR
1E AB 00 90 0529 1088 -MOVB #RAB$C_SEQ,RAB$B_RAC(R8); Reset to sequential access mode
OF 50 E9 052D 1090 BLBC R0,20$ ; Error if LBC
0530 1091 $GET RAB = R8, - ; Advance past this record which has
0530 1092 ERR = SUM$READ_ERR ; read before.
053F 1093 20$:
05 053F 1094 RSB
```

ACCESS_UPDATE

```
0540 1096 .SBTTL ACCESS_UPDATE
0540 1097
0540 1098 Routine to access update file
0540 1099
0540 1100
0540 1101 Inputs:
0540 1102
0540 1103 R8 = Main program RAB address
0540 1104 R10 = File block address of required update file
0540 1105 R11 = Edit block address of next edit
0540 1106
0540 1107
0540 1108 Outputs:
0540 1109
0540 1110 R9 = FAB address
0540 1111
0540 1112
0540 1113 ACCESS_UPDATE:
0540 1114 PUSH R2,R3,R4
0540 1115 IS_L_OPEN_FILE(R9),R2 ; Set pointer to file open
0540 1116 IS_L_CONN_FILE(R9),R3 ; and file connected markers
0540 1117 IS_T_FAB(R9),R4 ; Set pointer to SUM's FAB
0540 1118 RAB$Q_ISI(R8) ; Is RAB connected to a FAB?
0540 1119 BEQL 30$ ; No if EQL
0540 1120 CMPL RAB$L_FAB(R8),R4 ; Is it connected to SUM's FAB?
0540 1121 BNEQ 10$ ; No if NEQ
0540 1122 CMPL R10,(R3) ; Is it connected to required file?
0540 1123 BEQL 40$ ; Yes if EQL
0540 1124 BRB 20$
0540 1125 10$:
0540 1126 BSBW SAVE_SRC_RFA ; Save source file RFA
0540 1127 20$:
0540 1128 $DISCONNECT RAB = R8, - ; Disconnect RAB from FAB
0540 1129 ERR = SUM$CLOSE_ERR
0540 1130 BLBC R0,50$ ; Error if LBC
0540 1131 CLRL (R3) ; Mark that no file is connected
0540 1132 CMPL R10,(R2) ; Is required file already open?
0540 1133 BEQL 30$ ; Yes if EQL
0540 1134 TSTL (R2) ; Is any file open on this FAB?
0540 1135 BEQL 25$ ; No if EQL
0540 1136 $CLOSE FAB = R4, - ; Close currently open update file
0540 1137 ERR = SUM$CLOSE_ERR
0540 1138 BLBC R0,50$ ; Error if LBC
0540 1139 CLRL (R2) ; Mark that no file is open
0540 1140 25$:
0540 1141 MOVAL UPF T_NAM(R10), - ; Put NAM block into FAB
0540 1142 FAB$L_NAM(R4)
0540 1143 $OPEN FAB = R4, - ; Open required update file
0540 1144 ERR = SUM$OPEN_ERR
0540 1145 BLBC R0,50$ ; Error if LBC
0540 1146 MOVL R10,(R2) ; Mark which file is open
0540 1147 30$:
0540 1148 MOVL R4,RAB$L_FAB(R8) ; Put FAB address in RAB
0540 1149 $CONNECT RAB = R8, - ; Connect RAB to FAB
0540 1150 ERR = SUM$OPEN_ERR
0540 1151 BLBC R0,50$ ; Error if LBC
0540 1152 MOVL R10,(R3) ; Mark which file is connected
```

52	08	1C	BB	0540	1114		
53	0C	A9	9E	0542	1115		
54	32	A9	DE	054A	1117		
	02	A8	B5	054E	1118		
		5B	13	0551	1119		
54	3C	A8	D1	0553	1120		
		07	12	0557	1121		
63		5A	D1	0559	1122		
		69	13	055C	1123		
		03	11	055E	1124		
				0560	1125	10\$:	
	FF	86	30	0560	1126		
				0563	1127	20\$:	
				0563	1128		
				0563	1129		
73	50		E9	0572	1130		
	63		D4	0575	1131		
62	5A		D1	0577	1132		
	32		13	057A	1133		
	62		D5	057C	1134		
	14		13	057E	1135		
				0580	1136		
				0580	1137		
56	50		E9	058F	1138		
	62		D4	0592	1139		
				0594	1140	25\$:	
28	A4	38	AA	DE	0594	1141	
				0599	1142		
				0599	1143		
				0599	1144		
	3D	50	E9	05AB	1145		
62	5A		D0	05AB	1146		
				05AE	1147	30\$:	
3C	A8	54	D0	05AE	1148		
				05B2	1149		
				05B2	1150		
	24	50	E9	05C1	1151		
63	5A		D0	05C4	1152		

ACCESS_UPDATE

10 AB	0E AB	D0	05C7	1153	405:		
			05C7	1154		MOVL	ED W RFA+0(R11), - ; Reset RFA (3 words)
			05CC	1155			RAB\$Q RFA+0(R8)
14 AB	12 AB	B0	05CC	1156		MOVW	ED W RFA+4(R11), -
			05D1	1157			RAB\$Q RFA+4(R8)
1E AB	02	90	05D1	1158		MOVB	#RAB\$C_RFA,RAB\$B_RAC(R8); Put into RFA access mode
			05D5	1159		\$FIND	RAB = R8, - ; Position file
			05D5	1160			ERR = SUM\$READ ERR
1E AB	00	90	05E4	1161		MOVB	#RAB\$C_SEQ,RAB\$B_RAC(R8); Reset to sequential access mode
			05E8	1162	505:		
	1C	BA	05E8	1163		POPR	#^M<R2,R3,R4>
		05	05EA	1164		RSB	


```
READ_SRC_LINE
05EB 1166      .SBTTL  READ_SRC_LINE
05EB 1167      :
05EB 1168      Routine to read one line from source file
05EB 1169      :
05EB 1170      There are two entry points:
05EB 1171      :
05EB 1172      READ_SRC_LINE  to access file and read line
05EB 1173      :
05EB 1174      READ_SRC_LINEA if file is already accessed and ready to
05EB 1175      read next line
05EB 1176      :
05EB 1177      Inputs:
05EB 1178      :
05EB 1179      R8  = RAB address
05EB 1180      :
05EB 1181      Outputs:
05EB 1182      :
05EB 1183      R0  = Success/error status
05EB 1184      R6  = Line size
05EB 1185      R7  = Line buffer address
05EB 1186      IS_W_LINE_NO(R9) = line number
05EB 1187      :
05EB 1188      .ENABL  LSB
05EB 1189      :
05EB 1190      READ_SRC_LINE:
05EB 1191      MOVL  #1,R0          ; Assume success
05EB 1192      BSBW  ACCESS_SRC      ; Access source file
05EB 1193      BLBC  R0,20$        ; Error if LBC
05EB 1194      :
05EB 1195      READ_SRC_LINEA:
05EB 1196      $GET  RAB = R8, -      ; Get next line from source file
05EB 1197      ERR = SUM$READ_ERR
05EB 1198      BLBS  R0,10$          ; OK if LBS
05EB 1199      CMPL  R0,#RM$$_EOF      ; Was error end-of-file?
05EB 1200      BNEQ  20$            ; No if NEQ
05EB 1201      MOVB  #SUM$_ST_EOF,IS_B_STATE(R9) ; Set into EOF state
05EB 1202      BRB   20$
05EB 1203      10$:
05EB 1204      MOVZWL RAB$_RSZ(R8),R6 ; Set record size
05EB 1205      MOVL  RAB$_RBF(R8),R7  ; and buffer address
05EB 1206      INCW  IS_W_LINE_NO(R9) ; Increment line number
05EB 1207      BICB2 #SUM$_M_SRCUPD,IS_B_FLAGS(R9) ; Mark as source line
05EB 1208      CLRW  IS_W_INSERT_NO(R9) ; Reset new/replacement lines count
05EB 1209      20$:
05EB 1210      RSB
05EB 1211      :
05EB 1212      .DSABL  LSB
```

50 01 D0 05EB 1191
FEBA 30 05EE 1192
33 50 E9 05F1 1193
05F4 1194
05F4 1195
05F4 1196
05F4 1197
OF 50 E8 0603 1198
0001827A 8F 50 D1 0606 1199
18 12 060D 1200
04 A9 08 90 060F 1201
12 11 0613 1202
56 22 A8 3C 0615 1203
57 28 A8 D0 0619 1204
06 A9 B6 061D 1205
2A A9 04 8A 0620 1206
2E A9 B4 0624 1207
05 0627 1208
0627 1209
0628 1210
0628 1211
0628 1212

SKIP_SRC_LINES

```
0628 1214 .SBTTL SKIP_SRC_LINES
0628 1215
0628 1216 Routine to skip over source file lines
0628 1217
0628 1218 Inputs:
0628 1219
0628 1220 R4 = Last line number to skip
0628 1221 R8 = RAB address
0628 1222
0628 1223 Outputs:
0628 1224
0628 1225 IS_W_LINE_NO(R9) = Last line number
0628 1226
0628 1227
0628 1228 SKIP_SRC_LINES:
0628 1229 MOVL #1,R0 ; Assume success
0628 1230 CMPW R4,IS_W_LINE_NO(R9) ; Need to skip any?
0628 1231 BLSS 20$ ; No if LSS
0628 1232 BSBW ACCESS_SRC ; Access source file
0628 1233 BLBC R0,20$ ; Error if LBC
0628 1234 10$:
0628 1235 $FIND RAB = R8, - ; Skip one line
0628 1236 ERR = SUMSREAD_ERR
0628 1237 BLBC R0,20$ ; Error if LBC
0628 1238 INCW IS_W_DELETES(R9) ; Increment deleted lines count
0628 1239 ACBW R4,#T,IS_W_LINE_NO(R9),10$ ; Increment line number and branch b
0628 1240 ; if more lines to skip
0628 1241 BBSS #PRC_V_DELINE, - ; Set deleted lines information
0628 1242 IS_B_PROCFLAGS(R9),20$ ; pending flag
0628 1243 20$:
0628 1244 RSB
```

50 01 D0 0628 1229
06 A9 54 B1 0628 1230
27 19 0628 1231
FE77 30 0628 1232
21 50 E9 0628 1233
OF 50 E9 0628 1234
30 A9 B6 0628 1235
FFE4 06 A9 01 54 3D 0628 1236
00 05 A9 01 E2 0628 1237
0628 1238
0628 1239
0628 1240
0628 1241
0628 1242
0628 1243
05 0628 1244

COMMAND_CHECK

```
0659 1246 .SBTTL COMMAND_CHECK
0659 1247
0659 1248 Subroutine to check if line is a command
0659 1249
0659 1250 Inputs:
0659 1251
0659 1252 R6 = Size of line
0659 1253 R7 = Address of line
0659 1254 R8 = RAB address
0659 1255 R9 = Input stream control block
0659 1256 R10 = File block address
0659 1257
0659 1258 Outputs:
0659 1259 R4[CMND] = 0:Data 1:Command
0659 1260 R4[EDTRM] = 0:Normal command 1:Data terminator command
0659 1261 R4[EDEND] = 0:Data terminator 1:End of edit
0659 1262 R6 = Size of line
0659 1263 R7 = Address of line
0659 1264
0659 1265 COMMAND_CHECK:
0659 1266 ASSUME UPF_W_LOC2 EQ <UPF_W_LOC1+2>
0659 1267 MOVL R8, -SOM_CUR_RAB ; Make the currently active RAB available
0659 1268 ; to the TPARSE action routines.
0659 1269 MOVAL TPARSE_BLOCK, R1 ; Set pointer to Tparse parameter block
0659 1270 MOV B, FFLAGS(R9), - ; Get current input stream flags byte
0659 1271 TPA B, ISFLAGS(R1)
0659 1272 BBCC #SOM_V_AUDITNEW, - ; but clear new audit trail flag
0659 1273 TPA B, ISFLAGS(R1), 5$
0659 1274 5$:
0659 1275 CLRB TPA_B_EDFLAGS(R1) ; Clear all edit flags
0659 1276 MOVW UPF_W_DOT(R10), - ; Get current dot value
0659 1277 TPA W, DOT(R1)
0659 1278 CLRL TPA_W_LOC(R1) ; Clear locator value and line type
0659 1279 CLRL TPA_W_LOC1(R1) ; Clear loc-1 and loc-2
0659 1280 CLRL TPA_Q_AUDDS(R1) ; Clear audit descriptor
0659 1281 CLRL TPA_Q_CMNT(R1) ; Comment descriptor
0659 1282 MOVQ R6, TPA_Q_LINEDS(R1) ; Save line size and address
0659 1283 MOVQ R6, TPA_W_STRINGCNT(R1) ; Set TPARSE input descriptor
0659 1284 PUSHAL MER_KEY
0659 1285 PUSHAL MER_STATE
0659 1286 PUSH R1
0659 1287 CALLS #3, G*LIB$TPARSE
0659 1288 BLBC R0, 20$ ; Error if LBC
0659 1289 MOVAL TPARSE_BLOCK, R1 ; Set pointer to Tparse parameter block
0659 1290 TSTW TPA_W_LOC2(R1) ; Were two locators in command?
0659 1291 BEQL 8$ ; No if EQL, so don't compare them
0659 1292 CMPW TPA_W_LOC1(R1), TPA_W_LOC2(R1) ; Is loc-1 <= loc-2?
0659 1293 BLEQ 8$ ; Yes if LEQ
0659 1294 CLRL R0 ; Set error status
0659 1295 BRB 20$ ; and return
0659 1296 8$:
0659 1297 MOVQ TPA_Q_LINEDS(R1), R6 ; Reset line size and address,
0659 1298 MOVW R6, RAB$W_RSZ(R8) ; Reset RAB block record size
0659 1299 MOV B, TPA_B_ISFLAGS(R1), - ; input stream flags byte,
0659 1300 IS B, FFLAGS(R9)
0659 1301 MOV B, TPA_B_EDFLAGS(R1), - ; edit flags byte,
0659 1302 UPF_B_EDFLAGS(R10)
```

00000000'EF 58 D0
51 00000008'EF DE
28 A1 2A A9 90
00 28 A1 01 E5
2A A1 29 A1 94
0A AA B0
2C A1 D4
24 A1 D4
30 A1 7C
38 A1 7C
40 A1 56 7D
08 A1 56 7D
00000000'EF DF
00000000'EF DF
51 DD
00000000'GF 03 FB
51 50 E9
51 00000008'EF DE
26 A1 B5
0B 13
26 A1 24 A1 B1
04 15
50 D4
3A 11
50 40 A1 7D
22 A8 56 B0
2A A9 28 A1 90
09 AA 29 A1 90

COMMAND_CHECK

0A AA	2A A1	B0	06CE	1303	
04 AA	24 A1	D0	06D3	1304	
			06DB	1305	
20 AA	38 A1	7D	06DB	1306	
			06DD	1307	
10 2A A9	01	E1	06DD	1308	
			06E2	1309	
18 AA	30 A1	D0	06E2	1310	
			06E7	1311	
	3F	BB	06E7	1312	
28 AA	34 B1	30 A1	28	06E9	1313
				06F0	1314
	3F	BA	06F0	1315	
			06F2	1316	10\$:
54	2E A1	3C	06F2	1317	
			06F6	1318	20\$:
		05	06F6	1319	

MOVW	TPA_W_DOT(R1),UPF_W_DOT(R10)	; dot value,
MOVL	TPA_W_LOC1(R1),-	; locator 1,
	UPF_W_LOC1(R10),-	; and locator 2
MOVQ	TPA_Q_CMNT(R1),-	; Comment descriptor
	UPF_Q_CMNT(R10),-	
BBC	#SUM_V_AUDITNEW,-	; If new audit trail
	IS_B_FLAGS(R9),10\$	
MOVL	TPA_Q_AUDDS(R1),-	; Copy size of string
	UPF_Q_AUDDS(R10),-	
PUSHR	#MZR0,R1,R2,R3,R4,R5>	
MOVCL	TPA_Q_AUDDS(R1),-	; Copy audit string
	@TPA_Q_AUDDS+4(R1),UPF_T_AUDST(R10)	
POPR	#M<R0,R1,R2,R3,R4,R5>-	
MOVZWL	TPA_W_LINTYP(R1),P4	; Set line type flags
RSB		

COMMAND_CHECK

```
06F7 1321 :  
06F7 1322 : Tparse action routines  
06F7 1323 :  
06F7 1324 :  
06F7 1325 ACT_BLANKS_SIG:  
00 04 AC 00 0000 06F7 1326 .WORD 0  
E2 06F9 1327 BBSS #TPASV_BLANKS,TPASL_OPTIONS(AP),10$  
04 06FE 1328 10$:  
06FE 1329 RET  
06FF 1330 :  
06FF 1331 :  
06FF 1332 ACT_BLANKS_NSIG:  
00 04 AC 00 0000 06FF 1333 .WORD 0  
E5 0701 1334 BBCC #TPASV_BLANKS,TPASL_OPTIONS(AP),10$  
04 0706 1335 10$:  
0706 1336 RET  
0707 1337 :  
0707 1338 :  
0707 1339 ACT_PERCENT:  
28 AC 01 0000 0707 1340 .WORD 0  
88 0709 1341 BISB #SUM_M_AUDIT, - ; Switch on audit trail  
04 070D 1342 RET  
070E 1343 :  
070E 1344 :  
070E 1345 :  
070E 1346 ACT_BACKSLASH:  
28 AC 01 0000 070E 1347 .WORD 0  
8A 0710 1348 BICB #SUM_M_AUDIT, - ; Switch off audit trail  
04 0714 1349 RET  
0714 1350 :  
0715 1351 :  
0715 1352 :  
0715 1353 ACT_ESC:  
51 01 0040 0715 1354 .WORD ^M<R6>  
56 40 AC 9E 0717 1355 MOVL #1,R1 ; Set index  
66 B7 071A 1356 MOVAB TPA_Q_LINEDS(AP),R6 ; Point to buffer descriptor  
3F BB 071E 1357 DECB (R6) ; Reduce line length by one  
04 B6 04 B641 66 BB 0720 1358 PUSHB #^M<R0,R1,R2,R3,R4,R5> ; Save registers across MOVC3s.  
51 00000000 EF 28 0722 1359 MOVC3 (R6),24(R6)[R1],24(R6) ; Move up line.  
10 E0 0729 1360 MOVL SUM_CUR_RAB, R1 ; Get the current RAB,  
06 04 A1 0730 1361 BBS #RABSV_LOC, - ; check to see if we should  
24 B1 04 B6 66 28 0732 1362 RABSL_ROP(R1), 10$ ; propagate the shifted string  
3F BA 073B 1363 MOVC3 (R6),24(R6),24(R6)UBF(R1) ; to the UBF.  
04 073D 1364 10$: POPB #^M<R0,R1,R2,R3,R4,R5> ; Restore registers.  
073E 1365 RET  
073E 1366 :  
073E 1367 :  
073E 1368 ACT_EXIT:  
2E AC 20 AC 0000 073E 1369 .WORD 0  
B0 0740 1370 MOVW TPASL_PARAM(AP), - ; Set return type  
04 0745 1371 RET  
0746 1372 :  
0746 1373 :  
0746 1374 :  
0746 1375 ACT_LOC1:  
2E AC 01 0000 0746 1376 .WORD 0  
B0 0748 1377 MOVW #CMD_M_CMND,TPA_W_LINTYP(AP) ; Assume normal command
```

```
COMMAND_CHECK
24 AC 2C AC B0 074C 1378 MOVW TPA_W_LOC(AP),TPA_W_LOC1(AP)
      07 13 0751 1379 BEQL 10$
      2E AC 03 B0 0753 1380 MOVW #CMD_M_CMND!CMD_M_EDTRM,TPA_W_LINTYP(AP) ; If EQL is a normal command
      2C AC B4 0757 1381 CLRW TPA_Q_LOC(AP) ; Set as data terminator co
      04 075A 1382 10$:
      075A 1383 RET
      075B 1394 :
      075B 1385 :
      075B 1386 ACT_LOC2:
      0000 075B 1387 .WORD 0
26 AC 2C AC B0 075D 1388 MOVW TPA_W_LOC(AP),TPA_W_LOC2(AP)
      04 0762 1389 RET
      0763 1390 :
      0763 1391 :
      0763 1392 ACT_DOT:
      0000 0763 1393 .WORD 0
2C AC 2A AC B0 0765 1394 MOVW TPA_W_DOT(AP),TPA_W_LOC(AP)
      04 076A 1395 RET
      076B 1396 :
      076B 1397 :
      076B 1398 ACT_LOCNUM:
      0000 076B 1399 .WORD 0
2C AC 1C AC B0 076D 1400 MOVW TPASL_NUMBER(AP),TPA_W_LOC(AP)
2A AC 2C AC B0 0772 1401 MOVW TPA_W_LOC(AP),TPA_W_DOT(AP)
      04 0777 1402 RET
      0778 1403 :
      0778 1404 :
      0778 1405 ACT_PLUS:
      0000 0778 1406 .WORD 0
2C AC 1C AC A0 077A 1407 ADDW2 TPASL_NUMBER(AP),TPA_W_LOC(AP)
2A AC 2C AC B0 077F 1408 MOVW TPA_W_LOC(AP),TPA_W_DOT(AP)
      04 0784 1409 RET
      0785 1410 :
      0785 1411 :
      0785 1412 ACT_AUDIT:
      0000 0785 1413 .WORD 0
34 AC 0C AC D0 0787 1414 MOVL TPASL_STRINGPTR(AP),TPA_Q_AUDDS+4(AP)
      28 AC 02 88 078C 1415 BISB #SUM_M_AUDITNEW, - ; Set new audit trail flag
      0790 1416 TPA_Q_ISFLAGS(AP)
00 04 AC 00 E2 0790 1417 BBSS #TPASV_BLANKS,TPASL_OPTIONS(AP),10$ ; Make blanks significant
      04 0795 1418 10$:
      0795 1419 RET
      0796 1420 :
      0796 1421 :
      0796 1422 ACT_AUDCH:
      0000 0796 1423 .WORD 0
      10 30 AC D1 0798 1424 CMPL TPA_Q_AUDDS(AP),#16 ; Is audit trail at maximum size?
      03 18 079C 1425 BGEQ 10$ ; Yes if GEQ
      30 AC D6 079E 1426 INCL TPA_Q_AUDDS(AP) ; Increment audit trail size
      04 07A1 1427 10$:
      07A1 1428 RET
      07A2 1429 :
      07A2 1430 :
      07A2 1431 ACT_AUDEND:
      0000 07A2 1432 .WORD 0
00 04 AC 00 E5 07A4 1433 BBCC #TPASV_BLANKS,TPASL_OPTIONS(AP),10$ ; Switch off blank processing
      07A9 1434 10$:
```

```
COMMAND_CHECK
04 07A9 1435 RET
07AA 1436 :
07AA 1437 :
07AA 1438 ACT_CMNT:
07AA 1439 .WORD 0
38 AC 08 AC 0000 07AC 1440 MOVQ TPA$L_STRINGCNT(AP),TPA_Q_CMNT(AP)
04 07B1 1441 RET
07B2 1442 :
07B2 1443 :
07B2 1444 ACT_SUPPRESS:
07B2 1445 .WORD 0
29 AC 01 88 07B4 1446 BISB #ED_M_SUPPRESS, - ; Set clash messages suppressed flag
07B8 1447 TPA_B_EDFLAGS(AP)
04 07B8 1448 RET
```

```
SUMSCLOSE
07B9 1450      .SBTTL SUMSCLOSE
07B9 1451      :
07B9 1452      :
07B9 1453      This procedure is called from the main program prior to closing
07B9 1454      the input file. It ensures that the main program source file
07B9 1455      is connected to the RAB.
07B9 1456      :
07B9 1457      :
07B9 1458      Inputs:
07B9 1459      :
07B9 1460      4(AP) = Address of SUM control block
07B9 1461      :
07B9 1462      Outputs:
07B9 1463      :
07B9 1464      None
07B9 1465      :
07B9 1466      :
0300 07B9 1467      .ENTRY SUMSCLOSE, *M(R8,R9)
07B8 1468      :
51 04 AC D0 07B8 1469      MOVL 4(AP),R1          ; Get control block address
59 04 A1 D0 07BF 1470      MOVL SUM_L_ISDATA(R1),R9      ; and set data block pointer
      2C 13 07C3 1471      BEQL 20$
      69 D5 07C5 1472      TSTL IS_L_FILELIST(R9)        ; Is there an update list?
      28 13 07C7 1473      BEQL 20$                    ; No if EQL, file already accessed
51 1C A9 D0 07C9 1474      MOVL IS_L_MAIN_FAB(R9),R1      ; Get main program FAB address
      02 A1 B5 07CD 1475      TSTW FAB$Q_IFI(R1)          ; Is source file open?
      07 13 07D0 1476      BEQL 10$                    ; No if EQL
58 20 A9 D0 07D2 1477      MOVL IS_L_MAIN_RAB(R9),R8      ; Set RAB pointer
      FCD2 30 07D6 1478      BSBW ACCESS_SRC             ; Access source file
      07D9 1479 10$:
      08 A9 D5 07D9 1480      TSTL IS_L_OPEN_FILE(R9)     ; Is an update file open?
      13 13 07DC 1481      BEQL 20$                    ; No if EQL
      07DE 1482      $CLOSE FAB = IS T FAB(R9), -
      07DE 1483      ERR = SUMSCLOSE ERR
      08 A9 D4 07EE 1484      CLRL IS_L_OPEN_FILE(R9)     ; Close update file
      07F1 1485 20$:
      04 07F1 1486      RET
      07F2 1487      :
      07F2 1488      :
      07F2 1489      .END
```


SUMSEDT
Symbol table

D 4

16-SEP-1984 02:10:14 VAX/VMS Macro V04-00
5-SEP-1984 03:38:52 [SUM.SRC]SUMEDIT.MAR;1

Page 41
(28)

```

$$$CNT      = 00000003
$$$FLG      = FFFFFFFF
$$$KEY      = FFFFFFFF
$$$KFG      = FFFFFFFF
$$$MOD      = 00000000
$$$.TMP1    = 00000002
$$$.TMP2    = 000000A9
$$KEYTAB    = 00000000 R    05
..AFLG      = 00000000
..FLG       = 00000002
..MOD       = 00000000
..TYP       = 0000001F
..LEN       = 00000001
ACCESS_SRC  = 0000C4AB R    07
ACCESS_UPDATE = 00000540 R    07
ACT_AUDCH   = 00000796 R    07
ACT_AUDEND  = 000007A2 R    07
ACT_AUDIT   = 00000785 R    07
ACT_BACKSLASH = 0000070E R    07
ACT_BLANKS_NSIG = 000006FF R    07
ACT_BLANKS_SIG = 000006F7 R    07
ACT_CMNT    = 000007AA R    07
ACT_DOT     = 00000763 R    07
ACT_ESC     = 00000715 R    07
ACT_EXIT    = 0000073E R    07
ACT_LOC1    = 00000746 R    07
ACT_LOC2    = 0000075B R    07
ACT_LOCNUM  = 0000076B R    07
ACT_PERCENT = 00000707 R    07
ACT_PLUS    = 00000778 R    07
ACT_SUPPRESS = 000007B2 R    07
AUDCH       = 000000A9 R    04
BIT...      = 00000005
CHECK_ERR   = 000003B0 R    07
CMD_M_ALL   = 00000007
CMD_M_CMND  = 00000001
CMD_M_EDEND = 00000004
CMD_M_EDTRM = 00000002
CMD_V_CMND  = 00C00000
CMD_V_EDEND = 00000002
CMD_V_EDTRM = 00000001
CMND        = 0000003F R    04
CMNT        = 000000C3 R    04
COMMA       = 0000002C
COMMAND_CHECK = 00000659 R    07
DATA        = 00000032 R    04
EDIT        = 00000059 R    04
ED_B_FILENO = 00000019
ED_B_FLAGS  = 00000018
ED_K_BLN    = 0000001A
ED_L_BWD    = 00000004
ED_L_FILE   = 00000014
ED_L_FWD    = 00000000
ED_M_SEQERR = 00000002
ED_M_SUPPRESS = 00000001
ED_V_SEQERR = 00000001
ED_V_SUPPRESS = 00000000

```

```

ED_W_LINES  = 0000000C
ED_W_LOC1   = 00000008
ED_W_LOC2   = 0000000A
ED_W_RFA    = 0000000E
FABS_BID    = 00000000
FABS_BLN    = 00000001
FABS_C_BID  = 00000003
FABS_C_BLN  = 00000050
FABS_K_BLN  = 00000050
FABS_L_FOP  = 00000004
FABS_L_NAM  = 00000028
FABS_V_NAM  = 00000018
FABS_W_IFI  = 00000002
GET_IS_BLK  = 000000BA R    07
INSERT_NODE = 000001F5 R    07
IS_B_FLAGS  = 0000002A
IS_B_PROCFLAGS = 00000005
IS_B_STATE  = 00000004
IS_K_BLN    = 00000082
IS_L_CONN_FILE = 0000000C
IS_L_EDIT_BLK = 00000010
IS_L_FILELIST = 00000000
IS_L_FIRST_EDIT = 00000014
IS_L_LAST_EDIT = 00000018
IS_L_MAIN_FAB = 0000001C
IS_L_MAIN_RAB = 00000020
IS_L_OPEN_FILE = 00000008
IS_T_FAB    = 00000032
IS_W_DELETES = 00000030
IS_W_HIGH_LOC2 = 0000002C
IS_W_INSERT_NO = 0000002E
IS_W_LINE_NO = 00000006
IS_W_MAIN_RFA = 00000024
LESSTHAN    = 0000003C
LIB$FREE_VM = ***** X    07
LIB$GET_VM  = ***** X    07
LIB$PARSE   = ***** X    07
LINE_BLK    = 0000040B R    07
LINE_EOF    = 000004A1 R    07
LINE_GET    = 0000042E R    07
LINE_NUP    = 00000327 R    07
LINE_SET    = 00000300 R    07
LINE_SRC    = 0000032D R    07
LINE_UPD    = 0000033E R    07
LINE_UPE    = 000003C4 R    07
LINE_UPR    = 000003EC R    07
LOCATOR     = 000000C7 R    04
MER_KEY     = 00000000 R    05
MER_STATE   = 00000000 R    04
NAM$B_RSL   = 00000003
NAM$K_BLN   = 00000060
NAM$L_RSA   = 00000004
PRC_M_DELINE = 00000002
PRC_M_ERRORS = 00000004
PRC_M_EXPED = 00000001
PRC_M_HIEDIT = 00000008
PRC_M_NODATA = 00000010

```

SUM
V04

001

SUMSEDT
Symbol table

E 4

16-SEP-1984 02:10:14 VAX/VMS Macro V04-00
5-SEP-1984 03:38:52 [SUM.SRC]SUMEDIT.MAR;1

Page 42
(28)

PRC_V_DELINE	=	00000001		
PRC_V_ERRORS	=	00000002		
PRC_V_EXPED	=	00000000		
PRC_V_HIEDIT	=	00000003		
PRC_V_NODATA	=	00000004		
PROCESS_FILE	=	000000EF	R	07
RABSB_RAC	=	0000001E		
RABSC_RFA	=	00000002		
RABSC_SEQ	=	00000000		
RABSL_FAB	=	0000003C		
RABSL_RBF	=	00000028		
RABSL_ROP	=	00000004		
RABSL_UBF	=	00000024		
RABSM_LOC	=	00010000		
RABSV_LOC	=	00000010		
RABSW_ISI	=	00000002		
RABSW_RFA	=	00000010		
RABSW_RSZ	=	00000022		
RABSW_USZ	=	00000020		
READ_SRC_LINE	=	000005EB	R	07
READ_SRC_LINEA	=	000005F4	R	07
READ_UPD_LINE	=	00000230	R	07
READ_UPD_LINEA	=	00000236	R	07
RESTORE_SRC_RFA	=	000004F4	R	07
RMS\$ EOF	=	0001827A		
SAVE_SRC_RFA	=	000004E9	R	07
SEMICOLON	=	0000003B		
SET_UP_NODES	=	0000015A	R	07
SIZ...	=	00000001		
SKIP_SRC_LINES	=	00000628	R	07
SUM\$CLOSE	=	000007B9	RG	07
SUM\$CLOSE_ERR	=	*****	X	07
SUM\$INIT	=	00000006	R	07
SUM\$INIT_CMND	=	00000004	RG	07
SUM\$INIT_EDIT	=	00000000	RG	07
SUM\$LIB_ERR	=	*****	X	07
SUM\$LINE	=	000002B6	RG	07
SUM\$OPEN_ERR	=	*****	X	07
SUM\$READ_ERR	=	*****	X	07
SUM\$VIRT_ADDR	=	*****	X	07
SUM\$EDIT\$CLSH	=	00848800		
SUM\$EDOUTSEQ	=	00848818		
SUM\$PRMEOF	=	00848810		
SUM\$SLPSYNERR	=	00848808		
SUM_B_FLAGS	=	0000001C		
SUM_COR_RAB	=	00000000	R	02
SUM_DISPATCH	=	00000298	R	07
SUM_EDSIZE	=	00000004	R	03
SUM_ISSZE	=	00000000	R	03
SUM_K_BLN	=	0000001D		
SUM_L_ISDATA	=	00000004		
SUM_L_STS	=	00000000		
SUM_M_AUDIT	=	00000001		
SUM_M_AUDITNEW	=	00000002		
SUM_M_DELETE	=	00000010		
SUM_M_SRCUPD	=	00000004		
SUM_M_SUBCLSH	=	00000008		

SUM_Q_AUDDS	=	00000008		
SUM_Q_FILESP	=	00000010		
SUM_RETURN	=	000002AF	R	07
SUM_ST_BLK	=	00000006		
SUM_ST_EOF	=	00000008		
SUM_ST_GET	=	00000007		
SUM_ST_NUP	=	00000001		
SUM_ST_SET	=	00000000		
SUM_ST_SRC	=	00000002		
SUM_ST_UPD	=	00000003		
SUM_ST_UPE	=	00000004		
SUM_ST_UPR	=	00000005		
SUM_TP\$RSE	=	0000002C	R	02
SUM_UBF_ADDR	=	00000004	R	02
SUM_V_AUDIT	=	00000000		
SUM_V_AUDITNEW	=	00000001		
SUM_V_DELETE	=	00000004		
SUM_V_SRCUPD	=	00000002		
SUM_V_SUBCLSH	=	00000003		
SUM_W_INSERT_NO	=	0000001A		
SUM_W_LINE_NO	=	00000018		
SYSS\$CLOSE	=	*****	GX	07
SYSS\$CONNECT	=	*****	GX	07
SYSS\$DISCONNECT	=	*****	GX	07
SYSS\$FIND	=	*****	GX	07
SYSS\$GET	=	*****	GX	07
SYSS\$OPEN	=	*****	GX	07
SYSS\$REWIND	=	*****	GX	07
TERM	=	0000004C	R	04
TPASK_COUNT0	=	00000008		
TPASK_LENGTH0	=	00000024		
TPASL_NUMBER	=	0000001C		
TPASL_OPTIONS	=	00000004		
TPASL_PARAM	=	00000020		
TPASL_STRINGCNT	=	00000008		
TPASL_STRINGPTR	=	0000000C		
TPASV_BLANKS	=	00000000		
TPAS_ALPHA	=	000001EE		
TPAS_ANY	=	000001ED		
TPAS_BLANK	=	000001F2		
TPAS_DECIMAL	=	000001F3		
TPAS_DIGIT	=	000001EF		
TPAS_EOS	=	000001F7		
TPAS_EXIT	=	FFFFFFFF		
TPAS_FAIL	=	FFFFFFFE		
TPAS_FILESPEC	=	000001EA		
TPAS_HEX	=	000001F5		
TPAS_IDENT	=	000001EC		
TPAS_KEYWORD	=	00000100		
TPAS_LAMBDA	=	000001F6		
TPAS_MAXKEY	=	000000DC		
TPAS_OCTAL	=	000001F4		
TPAS_STRING	=	000001F0		
TPAS_SUBXPR	=	000001F8		
TPAS_SYMBOL	=	000001F1		
TPAS_UIC	=	000001EB		
TPARSE_BLOCK	=	00000008	R	02

SUMS
Syml

FAB
FAB
FAB
FAB
FAB
LIB
NAM
NAM
NAM
NAM
PUT
RAB
RAB
RMS
RMS
SHR
SHR
SHR
SHR
STS
STS
STS
STS
SUM
SUM
SUM
SUM
SUM
SUM
SYS

PSEC

SAB
SUM
SUM
SUM

Pha
Int
Com
Pas
Sym
Pas

SUM\$EDIT
Symbol table

TPA_B_EDFLAGS	=	00000029
TPA_B_ISFLAGS	=	00000028
TPA_Q_AUDDS	=	00000030
TPA_Q_CMNT	=	00000038
TPA_Q_LINEDS	=	00000040
TPA_W_DOT	=	0000002A
TPA_W_LINTYP	=	0000002E
TPA_W_LOC	=	0000002C
TPA_W_LOC1	=	00000024
TPA_W_LOC2	=	00000026
UPF_B_EDFLAGS		00000009
UPF_B_FIFLAGS		00000008
UPF_B_FILENO		0000000C
UPF_K_BLN		00000098
UPF_L_PTR		000000C0
UPF_Q_AUDDS		00000018
UPF_Q_CMNT		00000020
UPF_Q_EDITS		00000010
UPF_T_AUDST		00000028
UPF_T_NAM		00000038
UPF_V_INIT	=	00000000
UPF_W_DOT		0000000A
UPF_W_LOC1		00000004
UPF_W_LOC2		00000006

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$AB\$\$	00000098 (152.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SUM\$RW_DATA	00000050 (80.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
SUM\$RO_DATA	00000008 (8.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
-LIB\$STATES	000000E7 (231.)	04 (4.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE
-LIB\$KEYOS	00000000 (0.)	05 (5.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC WORD
-LIB\$KEY1\$	00000000 (0.)	06 (6.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC WORD
SUM\$CODE	000007F2 (2034.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	38	00:00:00.08	00:00:00.60
Command processing	147	00:00:00.51	00:00:01.72
Pass 1	476	00:00:23.82	00:00:49.76
Symbol table sort	0	00:00:01.18	00:00:01.99
Pass 2	254	00:00:05.73	00:00:11.81
Symbol table output	30	00:00:00.23	00:00:00.39
Psect synopsis output	3	00:00:00.04	00:00:00.06
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	950	00:00:31.60	00:01:06.56

The working set limit was 1950 pages.

F 4

16-SEP-1984 02:10:14 VAX/VMS Macro V04-00
5-SEP-1984 03:38:52 [SUM.SRC]SUM\$EDIT.MAR;1

Page 43
(28)

SUM\$
VAX-

Sym
Psect
Cross
Assoc

The
332
The
320
14

Mac

\$2
-2
TOT

688

The

MAC

121870 bytes (239 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 921 non-local and 83 local symbols.
1489 source lines were read in Pass 1, producing 43 object records in Pass 2.
65 pages of virtual memory were used to define 52 macros.

! Macro library statistics !

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[SUM.OBJ]SUM.MLB;1	7
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	29
TOTALS (all libraries)	36

1413 GETS were required to define 36 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LISS:SUMEDIT/OBJ=OBJ\$:SUMEDIT MSRC\$:SUMEDIT/UPDATE=(ENH\$:SUMEDIT)+LIB\$:SUM/LIB

0368 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

0369

AH-BT13A-SE
VAX/VMS V4.0

**DIGITAL
CONFIDE**

EQUIPMENT
TIAL AND

CORPORATION
PROPRIETARY